

Hello, and welcome to my personal website! Here, you can learn more about who I am, my work, and what I'm thinking about. In this summary, I'll take you through step-by-step how I created each webpage, challenges I encountered and how I dealt with them, and my favorite part of each webpage.

## Design

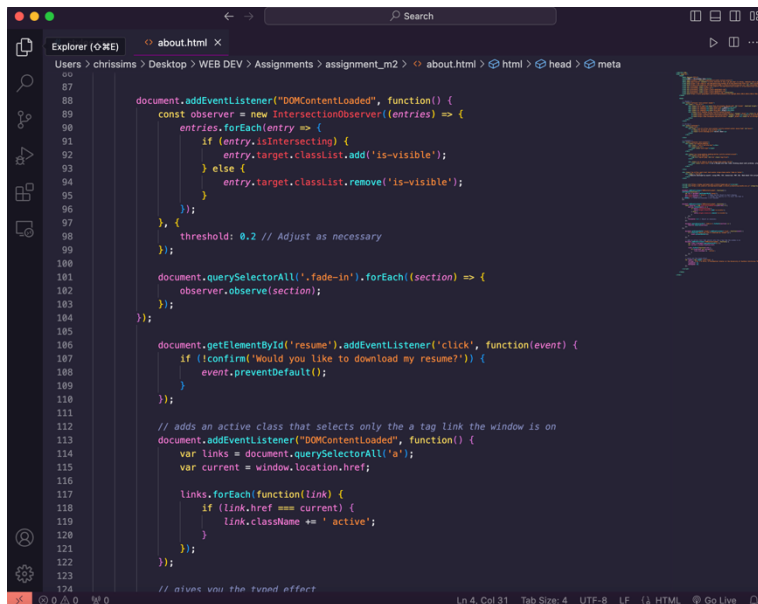
The design largely was inspired by a VSCode theme (plug-ins you can install into that changes the colors when you code), called Synthwave '84. Recently, I've been obsessed with the synthwave music genre, a type of electronic music originating in France made for video games and movies.

This music, although originating in the 2000s, attempts to inspire nostalgia for the 1980s. Although I was born way after the 1980s, I can't help but feel a sense of wonder and longing when I listen to the music (my favorite song being "Resonance" by Home). I attempted to capture the same feeling in my website.

Listen: <https://www.youtube.com/watch?v=8GW6sLrK40k>

I wanted the user to feel like they were in my code editor, which is why my font throughout the website is "Source Code Pro", and my sections are syntactically written as functions.

This is Synthwave '84, my VSCode theme :)



```
87
88 document.addEventListener("DOMContentLoaded", function() {
89     const observer = new IntersectionObserver(entries => {
90         entries.forEach(entry => {
91             if (entry.isIntersecting) {
92                 entry.target.classList.add('is-visible');
93             } else {
94                 entry.target.classList.remove('is-visible');
95             }
96         });
97     }, {
98         threshold: 0.2 // Adjust as necessary
99     });
100
101     document.querySelectorAll('.fade-in').forEach(section => {
102         observer.observe(section);
103     });
104 });
105
106 document.getElementById('resume').addEventListener('click', function(event) {
107     if (!confirm('Would you like to download my resume?')) {
108         event.preventDefault();
109     }
110 });
111
112 // adds an active class that selects only the a tag link the window is on
113 document.addEventListener("DOMContentLoaded", function() {
114     var links = document.querySelectorAll('a');
115     var current = window.location.href;
116
117     links.forEach(function(link) {
118         if (link.href === current) {
119             link.className += ' active';
120         }
121     });
122 });
123
124 // always you the tuned effect
```

## About Page

From the beginning, I knew I wanted my home page to be the most visual webpage. I wanted users of my website to feel like the home page was interacting with them, rather than vice versa. Thus, to add interactivity, I incorporated three features:

### 1) Typing Animation

I incorporated an open source, GitHub JavaScript library, Typed.js, to convey the feeling to users that I was actively typing out my words for them to see, rather than to have the words statically on the page. It was relatively easy, all I had to do was go through the README, watch a couple demos, and I had learned how to incorporate the typing animation.

Shout out: <https://github.com/mattboldt/typed.js>

### 2) Scroll Down Timer

I used JavaScript to time the “Scroll Down” mouse icon to appear when the typing animation finished. Also relatively easy, I physically counted the time it took for my animation to finish and changed the opacity and display after the animation finished.

### 3) Fade In Effect

Although subtle, all the features on my homepage fade-in as the user scrolls down the webpage. I wanted to add more complex parallax scrolling methods, and maybe I will in the future, by for now I was satisfied with using the IntersectionObserver function.

More than these features, the biggest challenge was getting my features positioned properly. I hadn’t used Bootstrap for positioning in a while before this and was rusty at first.

## Work Page

Here, I wanted to focus more on the content, than to confuse my users with complex visual techniques. So, I kept it simple – the three sections and the webpages they connect to is all coded in basic HTML and Bootstrap CSS.

The mail contact system was where I experienced most of my frustration. While initially creating the form and the JavaScript (using the mail() function) was relatively straightforward. However, the emails simply wouldn’t send and I had no clue why.

So I troubleshooted. Eventually I figured out that the local server environment I was using, MAMP, didn’t have a built-in SMTP for MacOS users. Thus, I went down a rabbit hole of learning about mail servers and eventually learned that I had to use a PHP library called PHPMailer to incorporate my mailing system.

Read more about it here: <https://github.com/PHPMailer/PHPMailer>

After reading the README and watching a couple demos, I was good to go.

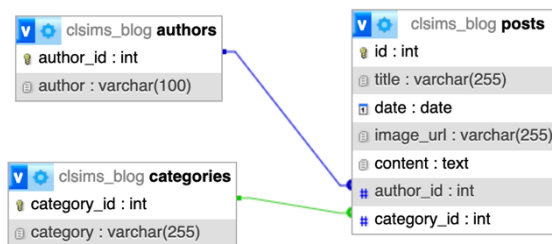
However, I also learned that my Gmail had a two-step authentication system that prevented me from directly inputting my username and password into PHPMailer. So I had to get around that too by going into my Google Settings and retrieving a one-time App Password.

Honestly, implementing the contact form was more trouble than it was worth. BUT I learned a surprising amount about random email-y things like routing, authentication, port usage, and encryption.

## Blog Page

Finally, the blog page which took up about 80% of my time on this website and where most of the backend is. After quickly styling how I wanted my posts to look, I started with first my database design, which I used phpMyAdmin, an admin tool made for MySQL (which I used to query and manage my database).

Here is a look at my database diagram.



My database was made up of ideas I had from my Notes app, with original dates.

I added three different tools so I could manage my posts directly on my website: a search tool, a create tool, and a delete tool. Here are some techniques I've used to accomplish this using PHP and SQL:

- 1) Dynamic Query Filters: based on user inputs, I dynamically append conditions to the SQL query allowing for customizable search.
- 2) Input handling and validation: checking for presence and non-emptiness of parameters before appending to SQL query.

- 3) Pagination: calculating total results, results per page, and other pagination logic to ensure users can navigate through pages of results
- 4) Password Hashing: for creating and deleting posts, I use a hashed password check for security and to authenticate actions.

#### EXTRAS

1. Event-driven DOM Manipulation
2. Pagination
3. Mail-to Function
4. Hashing
5. Responsive Web Design